Função obs.sim

```
obs.sim <-
function(obs,sim,indice=c("all","r2","r","d","c","EM","EAM","REQM","EF"))
#1 Cria função "obs.sim" com os argumentos "obs", "sim" e "indice".
{

##############################
##  Verificando as entradas  ##
##############################

if(any(is.na(obs) == TRUE))                                          #2
Verificando se existe dados faltantes no obs.
{obs <- na.omit(obs)}                                                #3
Se houver, exclui a(s) linha(s) em que houver dado(s) faltante(s).

if(any(is.na(sim) == TRUE))                                          #4
Verificando se existe dados faltantes no sim.
{sim <- na.omit(sim)}                                                #5
Se houver, exclui a(s) linha(s) em que houver dado(s) faltante(s.

if(class(obs) != "numeric" && "obs" >= 0)                            #6
Verificando o tipo de dado de entrada.
{ stop("obs precisa ser número e ≥ 0.") }                            #7
Caso não seja um numero ou seja menor que zero, retorna uma mensagem de
erro.

if(class(sim) != "data.frame" && "sim" >= 0)                         #8
Verificando o tipo de dado de entrada.
{ stop("sim precisa ser da classe data.frame e ≥ 0.") }              #9
Caso não seja um data.frame ou seja menor que zero, retorna uma mensagem de
erro.

if(length(obs) != nrow(sim))                                         #10
Verificando se os dados obs e sim apresentam mesma quantidade de valores.
{ stop("obs e sim devem apresentar mesma quantidade de dados.") }    #11
Caso obs e sim nao apresentem mesma quantidade de valores, retorna uma
mensagem de erro.

if(indice != "all" && indice != "r2" && indice != "r" && indice != "d" &&
indice != "c"     #12 Verificando se houve erro de digitacao.
   && indice != "EM" && indice != "EAM" && indice != "REQM" && indice !=
"EF")              #13 Verificando se houve erro de digitacao.
{ stop("indice pode ser all, r2, r, d, c, EM, EAM, REQM ou EF.") }
#14 Caso haja erro de digitacao, retorna uma mensagem de erro.

##########################################
##  Criando os parametros condicionais  ###
##########################################
```

```
if(indice == "all")                                                    #15
Se o usuario escolher todos os índices ("all").
  {
  N <-  ncol(sim)                                                      #16
Atribui a "N" o numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=10)                             #17
Cria uma matriz "resultado" para inserir os valores obtidos no fluxo ou
loop.
  rownames(resultado) <- c("r2","r","Interpretacao r","d","c",         #18
Atribui os nomes dos indices as linhas de "resultado".
                           "Desempenho c","EM","EAM","REQM","EF")      #19
Atribui os nomes dos indices as linhas de "resultado".
  colnames(resultado) = c(colnames(sim))                               #20
Atribui os nomes das colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                                                #21
Entra em um loop "for" com contador "i" de 1 até o número maximo de colunas
de "sim".
  {
    r2 <- round((summary((lm(sim[,i]~obs)))$r.squared),3)
#22 Atribui a "r2" o "r²" obtido no summary() da regressao linear.
    r <- round(cov(obs,sim[,i])/(sd(obs)*sd(sim[,i])),3)
#23 Atribui a "r" o resultado do calculo do índice estatístico "r".
    r.abs <- abs(r)
#24 Transforma os valores obtidos do índice estatistico "r" em valores
absolutos "r.abs".
    if(r.abs >= 0.0 && r.abs <= 0.1) {nivelr <- ("Muito baixa");}       #25
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Muito baixa".
    if(r.abs > 0.1 && r.abs <= 0.3) {nivelr <- ("Baixa");}              #26
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Baixa".
    if(r.abs > 0.3 && r.abs <= 0.5) {nivelr <- ("Moderada");}           #27
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Moderada".
    if(r.abs > 0.5 && r.abs <= 0.7) {nivelr <- ("Alta");}               #28
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Alta".
    if(r.abs > 0.7 && r.abs <= 0.9) {nivelr <- ("Muito alta");}         #29
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Muito alta".
    if(r.abs > 0.9 && r.abs <= 1.0) {nivelr <- ("Quase perfeita")}      #30
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Quase perfeita".
    d <- round(1-(sum((sim[,i]-obs)^2)/sum((abs(sim[,i]-mean(obs))+abs(obs-
mean(obs)))^2)),3)    #31 Atribui a "d" o resultado do calculo do índice
estatistico "d".
    c <- round((1-(sum((sim[,i]-obs)^2)/sum((abs(sim[,i]-mean(obs))+abs(obs-
mean(obs)))^2)))*    #32 Atribui a "c" o resultado do cálculo índice
estatistico "c".
                 sqrt(sum((sim[,i]-mean(obs))^2)/sum((obs-mean(obs))^2)),3)
```

```
#33 Atribui a "c" o resultado do cálculo índice estatistico "c".
    if(c <= 0.41) {nivelc <- ("Pessimo");}                              #34
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Pessimo".
    if(c > 0.41 && c <= 0.50) {nivelc <- ("Mau");}                      #35
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Mau".
    if(c > 0.51 && c <= 0.61) {nivelc <- ("Sofrivel");}                 #36
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Sofrivel".
    if(c > 0.61 && c <= 0.66) {nivelc <- ("Mediano");}                  #37
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Mediano".
    if(c > 0.66 && c <= 0.76) {nivelc <- ("Bom");}                      #38
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Bom".
    if(c > 0.76 && c <= 0.85) {nivelc <- ("Muito bom")};                #39
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Muito bom".
    if(c > 0.85) {nivelc <- ("Otimo")}                                  #40
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Otimo".
    EM <- round((sum(sim[,i]-obs))/length(obs),3)
#41 Atribui a "EM" o resultado do cálculo do indice estatistico "EM".
    EAM <- round((sum(abs(sim[,i]-obs)))/length(obs),3)
#42 Atribui a "EAM" o resultado do cálculo indice estatistico "EAM".
    REQM <- round(sqrt(sum((sim[,i]-obs)^2))/length(obs),3)
#43 Atribui a "REQM" o resultado do cálculo indice estatistico "REQM".
    EF <- round(1-((sum((obs-mean(obs))^2)-sum((obs-sim[,i])^2))/sum((obs-
mean(obs))^2)),3)     #44 Atribui a "EF" o resultado do cálculo indice
estatistico "EF".
    resultado[1,i] <- c(r2)                             #45 Atribui a linha 1
de "resultado" o(s) valor(es) de "r2".
    resultado[2,i] <- c(r)                              #46 Atribui a linha 2
de "resultado" o(s) valor(es) de "r".
    resultado[3,i] <- c(nivelr)                         #47 Atribui a linha 3
de "resultado" o(s) valor(es) "nivelr".
    resultado[4,i] <- c(d)                              #48 Atribui a linha 4
de "resultado" o(s) valor(es) de "d".
    resultado[5,i] <- c(c)                              #49 Atribui a linha 5
de "resultado" o(s) valor(es) de "c".
    resultado[6,i] <- c(nivelc)                         #50 Atribui a linha 6
de "resultado" o(s) valor(es) de "nivelc".
    resultado[7,i] <- c(EM)                             #51 Atribui a linha 7
de "resultado" o(s) valor(es) de "EM".
    resultado[8,i] <- c(EAM)                            #52 Atribui a linha 8
de "resultado" o(s) valor(es) de "EAM".
    resultado[9,i] <- c(REQM)                           #53 Atribui a linha 9
de "resultado" o(s) valor(es) de "REQM".
    resultado[10,i] <- c(EF)                            #54 Atribui a linha
10 de "resultado" o(s) valor(es) de "EF".
```

```
  }
  resultado <- as.data.frame(resultado)            #55 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "r2")                                 #56 Se o usuario escolher
somente o indice "r2".
{
  N <-  ncol(sim)                                  #57 Atribui a "N" o
numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=1)          #58 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("r2")                   #59 Atribui o nome do
indices "r2" a linha de "resultado".
  colnames(resultado) = c(colnames(sim))           #60 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                            #61 Entra em um loop
"for" com contador "i" de 1 ate o numero maximo de colunas de "sim".
  {
    r2 <- round((summary((lm(sim[,i]~obs)))$r.squared),3)      #62 Atribui a
"r2" o "r²" obtido no summary() da regressão linear.
    resultado[1,i] <- (r2)                         #63 Atribui a linha 1 de
"resultado" o(s) valor(es) de "r2".
  }
  resultado <- as.data.frame(resultado)            #64 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "r")                                  #65 Se o usuario escolher
somente o indice "r".
{
  N <-  ncol(sim)                                  #66 Atribui a "N" o
numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=2)          #67 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("r","Interpretacao r")  #68 Atribui o nome do
indice "r" e sua interpretacao as linhas de "resultado".
  colnames(resultado) = c(colnames(sim))           #69 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                            #70 Entra em um loop
"for" com contador "i" de 1 até o numero maximo de colunas de "sim".
  {
    r <- round(cov(obs,sim[,i])/(sd(obs)*sd(sim[,i])),3)               #71
Atribui a "r" o resultado do cálculo do indice estatistico "r".
    r.abs <- abs(r)                                                    #72
Transforma os valores obtidos do indice estatistico "r" em valores absolutos
"r.abs".
    if(r.abs >= 0.0 && r.abs <= 0.1) {nivelr <- ("Muito baixa");}      #73
```

```
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Muito baixa".
    if(r.abs > 0.1 && r.abs <= 0.3) {nivelr <- ("Baixa");}              #74
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Baixa".
    if(r.abs > 0.3 && r.abs <= 0.5) {nivelr <- ("Moderada");}           #75
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Moderada".
    if(r.abs > 0.5 && r.abs <= 0.7) {nivelr <- ("Alta");}               #76
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Alta".
    if(r.abs > 0.7 && r.abs <= 0.9) {nivelr <- ("Muito alta");}         #77
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Muito alta".
    if(r.abs > 0.9 && r.abs <= 1.0) {nivelr <- ("Quase perfeita")}      #78
Se o valor de "r.abs" estiver nesse intervalo, atribui a "nivelr" a
interpretacao "Quase perfeita".
    resultado[1,i] <- c(r)                          #79 Atribui a linha 1 de
"resultado" o(s) valor(es) de "r".
    resultado[2,i] <- c(nivelr)                     #80 Atribui a linha 2 de
"resultado" o(s) valor(es) de "nivelr".
  }
  resultado <- as.data.frame(resultado)             #81 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "d")                                   #82 Se o usuario escolher
somente o indice "d".
{
  N <-  ncol(sim)                                   #83 Atribui a "N" o
numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=1)           #84 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("d")                     #85 Atribui o nome do
indice "d" a linha de "resultado".
  colnames(resultado) = c(colnames(sim))            #86 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                             #87 Entra em um loop
"for" com contador "i" de 1 até o numero maximo de colunas de "sim".
  {
    d <- round(1-(sum((sim[,i]-obs)^2)/sum((abs(sim[,i]-mean(obs))+abs(obs-
mean(obs)))^2)),3)      #88 Atribui a "d" o resultado do calculo do indice
estatistico "d".
    resultado[1,i] <- c(d)                          #89 Atribui a linha 1 de
"resultado" o(s) valor(es) de "d".
  }
  resultado <- as.data.frame(resultado)             #90 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}
```

```
if(indice == "c")                                   #91 Se o usuario escolher
somente o índice "c".
{
  N <-  ncol(sim)                                   #92 Atribui a "N" o
número de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=2)           #93 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("c","Desempenho c")      #94 Atribui o nome do
indice "c" e seu desempenho as linhas de "resultado".
  colnames(resultado) = c(colnames(sim))            #95 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                             #96 Entra em um loop
"for" com contador "i" de 1 até o numero maximo de colunas de "sim".
  {
    c <- round((1-(sum((sim[,i]-obs)^2)/sum((abs(sim[,i]-mean(obs))+abs(obs-
mean(obs)))^2)))*sqrt(sum((sim[,i]-mean(obs))^2)/sum((obs-mean(obs))^2)),3)
#97 Atribui a "c" o resultado do cálculo do índice estatistico "c".
    if(c <= 0.41) {nivelc <- ("Pessimo");}                                #98
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Pessimo".
    if(c > 0.41 && c <= 0.50) {nivelc <- ("Mau");}                        #99
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Mau".
    if(c > 0.51 && c <= 0.61) {nivelc <- ("Sofrivel");}                   #100
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Sofrivel".
    if(c > 0.61 && c <= 0.66) {nivelc <- ("Mediano");}                    #101
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Mediano".
    if(c > 0.66 && c <= 0.76) {nivelc <- ("Bom");}                        #102
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Bom".
    if(c > 0.76 && c <= 0.85) {nivelc <- ("Muito bom")};                  #103
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Muito bom".
    if(c > 0.85) {nivelc <- ("Otimo")}                                    #104
Se o valor de "c" estiver nesse intervalo, atribui a "nivelc" o desempenho
"Otimo".
    resultado[1,i] <- c(c)                           #105 Atribui a linha 1
de "resultado" o(s) valor(es) de "c".
    resultado[2,i] <- c(nivelc)                      #106 Atribui alinha 2 de
"resultado" o(s) valor(es) de "nivelc".
  }
  resultado <- as.data.frame(resultado)              #107 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "EM")                                  #108 Se o usuario
escolher somente o indice "EM".
```

```
{
  N <-  ncol(sim)                                   #109 Atribui a "N" o
numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=1)           #110 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("EM")                    #111 Atribui o nome do
indice "EM" a linha de "resultado".
  colnames(resultado) = c(colnames(sim))            #112 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                             #113 Entra em um loop
"for" com contador "i" de 1 ate o numero maximo de colunas de "sim".
  {
    EM <- round((sum(sim[,i]-obs))/length(obs),3)     #114 Atribui a "EM" o
resultado do calculo do indice estatistico "EM".
    resultado[1,i] <- c(EM)                           #115 Atribui a linha 1
de "resultado" o(s) valor(es) de "EM".
  }
  resultado <- as.data.frame(resultado)             #116 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "EAM")                                 #117 Se o usuario
escolher somente o indice "EAM".
{
  N <-  ncol(sim)                                  #118 Atribui a "N" o
numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=1)          #119 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("EAM")                  #120 Atribui o nome do
indice "EAM" a linha de "resultado".
  colnames(resultado) = c(colnames(sim))           #121 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                            #122 Entra em um loop
"for" com contador "i" de 1 ate o numero maximo de colunas de "sim".
  {
    EAM <- round((sum(abs(sim[,i]-obs)))/length(obs),3)     #123 Atribui a
"EAM" o resultado do calculo do índice estatistico "EAM".
    resultado[1,i] <- c(EAM)                              #124 Atribui a
linha 1 de "resultado" o(s) valor(es) de "EAM".
  }
  resultado <- as.data.frame(resultado)            #125 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "REQM")                               #126 Se o usuario escolher
somente o índice "REQM".
{
  N <-  ncol(sim)                                  #127 Atribui a "N" o
numero de colunas de "sim".
```

```
  resultado <- matrix(NA, ncol=N, nrow=1)         #128 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("REQM")                #129 Atribui o nome do
indice 'REQM" a linha de "resultado".
  colnames(resultado) = c(colnames(sim))          #130 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                           #131 Entra em um loop
"for" com contador "i" de 1 ate o numero maximo de colunas de "sim".
  {
    REQM <- round(sqrt(sum((sim[,i]-obs)^2))/length(obs),3)    #132 Atribui
a "REQM" o resultado do cálculo do índice estatistico "REQM".
    resultado[1,i] <- c(REQM)                             #133 Atribui a
linha 1 de "resultado" o(s) valor(es) de "REQM".
  }
  resultado <- as.data.frame(resultado)           #134 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


if(indice == "EF")                                #135 Se o usuario escolher
somente o índice "EF".
{
  N <-  ncol(sim)                                 #136 Atribui a "N" o
numero de colunas de "sim".
  resultado <- matrix(NA, ncol=N, nrow=1)         #137 Cria uma matriz
"resultado" para inserir os valores obtidos no fluxo ou loop.
  rownames(resultado) <- c("EF")                  #138 Atribui o nome do
indice "EF" a linha de "resultado".
  colnames(resultado) = c(colnames(sim))          #139 Atribui os nomes das
colunas de "sim" as colunas de "resultado".
  for(i in 1:ncol(sim))                           #140 Entra em um loop
"for" com contador "i" de 1 ate o numero maximo de colunas de "sim".
  {
    EF <- round(1-((sum((obs-mean(obs))^2)-sum((obs-sim[,i])^2))/sum((obs-
mean(obs))^2)),3)     #141 Atribui a "EF" resultado do calculo do índice
estatistico "EF".
    resultado[1,i] <- c(EF)                       #142 Atribui a linha 1 de
"resultado" o(s) valor(es) de "EF".
  }
  resultado <- as.data.frame(resultado)           #143 Transforma
"resultado" em um "data.frame" e guarda em "resultado".
}


##################################
###   Criando os resultados   ###
##################################

### Tabela com os índices:
tabela <- resultado                  #144 Atribui a "tabela" o "data.frame"
```

```
"resultado.
View(tabela)                        #145 Abre uma nova aba no R com a
"tabela" organizada.

### Gráficos de dispersão:
for(j in 1:ncol(sim))                                          #146 Entra em
um loop "for" com contador "j" de 1 ate o numero maximo de colunas de "sim".
{
  max.obs <- max(obs, na.rm = TRUE)                            #147 Atribui a
"max.obs" o maior valor dos dados "obs".
  max.sim <- max(sim[,j], na.rm = TRUE)                        #148 Atribui a
"max.sim" o maior valor dos dados "sim".
  if (max.obs > max.sim)                                       #149 Se
"max.obs" for maior que "max.sim":
  {xy <- round(coef(lm(sim[,j]~obs)),2)                        #150 Atribui a
"xy" os coeficientes da regressão linear.
  x11()                                                        #151 Abre
dispositivo de tela.
  par (mfrow = c (1, 1))                                       #152
Dispositivo de tela sera para so uma coluna/um grafico.
  plot (sim[,j]~obs,                                           #153 Plota area
do grafico com os dados observados e simulados.
        bty = "o",                                             #154 Coloca
margens nos lados 1 e 2.
        pch = 19,                                              #155 Altera
formato de pontos para categoria 19.
        cex = 1.2,                                             #156 Altera o
tamanho dos pontos.
        xlab = "Observados", ylab = "Simulados",               #157 Insere os
titulos dos eixos x e y.
        xlim = c(0,max.obs), ylim = c(0,max.obs))              #158 Coloca os
titulos dos eixos x e y.
  abline(xy,lwd=3, col="red")                                     #159 Traca a
reta da regressao linear realizada "xy".
  segments(0,0,x1=max.obs,y1=max.obs, lwd=1.8, col = "black")    #160 Traca
a reta 1:1 no grafico de dispersao.
  eq <- paste0("obs = ", xy[1], ifelse(sign(xy[2])==1, " + ", " - "),
abs(xy[2]), " * sim")   #161 Atribui a "eq" os caracteres concatenados que
formam a equacao da regressao linear.
  mtext(eq, 1, line=-1, padj=0)
#162 Insere o "eq" no grafico correspondente.
  gr.r2 <- paste0("r2 = ", round((summary((lm(sim[,j]~obs)))$r.squared),3))
#163 Atribui a ""gr.r2" os caracteres: "r2" e seu valor correspondente.
  mtext(gr.r2, 1, line=-2, padj=0)}
#164 Insere o "gr.r2" no grafico correspondente.
  if (max.sim > max.obs)                                       #165 Se
"max.sim" for maior que "max.obs":
  {xy <- round(coef(lm(sim[,j]~obs)),2)                        #166 Atribui a
"xy" os coeficientes da regressao linear.
  x11()                                                        #167 Abre
dispositivo de tela.
```

```
  par (mfrow = c (1, 1))                                        #168
Dispositivo de tela sera para so uma coluna/um grafico.
  plot (sim[,j]~obs,                                            #169 Plota area
do grafico com os dados observados e simulados.
        bty = "o",                                              #170 Coloca
margens nos lados 1 e 2.
        pch = 19,                                               #171 Altera
formato de pontos para categoria 19.
        cex = 1.2,                                              #172 Altera o
tamanho dos pontos.
        xlab = "Observados", ylab = "Simulados",                #173 Insere os
titulos dos eixos x e y.
        xlim = c(0,max.sim), ylim = c(0,max.sim))               #174 Coloca os
titulos dos eixos x e y.
  abline(xy,lwd=3, col="red")                                           #175
Traca a reta da regressao linear realizada "xy".
  segments(0,0,x1=max.sim,y1=max.sim, lwd=1.8, col = "black")      #176
Traca a reta 1:1 no grafico de dispersao.
  eq <- paste0("obs = ", xy[1], ifelse(sign(xy[2])==1, " + ", " - "),
abs(xy[2]), " * sim")   #177 Atribui a "eq" os caracteres concatenados que
formam a equacao da regressao linear.
  mtext(eq, 1, line=-1, padj=0)
#178 Insere o "eq" no grafico correspondente.
  gr.r2 <- paste0("r2 = ", round((summary((lm(sim[,j]~obs)))$r.squared),3))
#179 Atribui a ""gr.r2" os caracteres: "r2" e seu valor correspondente.
  mtext(gr.r2, 1, line=-2, padj=0)}
#180 Insere o "gr.r2" no grafico correspondente.
}
  return(tabela)     #181 Retorna ao usuário a "tabela" com os resultados
dos indices estatisticos para cada     coluna de "sim".
}
```

From:
http://ecor.ib.usp.br/ - **ecoR**

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2018:alunos:trabalho_final:paola.bongiovani:funcao_obs.sim

Last update: **2020/08/12 06:04**